TEK4030, oblig2

fridamei

November 2021

1 Task 1

1.1 A

DH-table:

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
1	a_2	0	0	θ_2

Where a_i is the link length and θ_i is the joint angle of the i-th joint

This gives the following transformation matrices:

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = A_1A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The end effector position is given by:

$$o_2 = T_2^0 \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1c_1 + a_2c_{12} \\ a_1s_1 + a_2s_{12} \\ 0 \end{bmatrix}$$

Finding the Jacobian:

$$J(q) = \begin{bmatrix} z_0^0 \times (o_2 - o_0) & z_1^0 \times (o_2 - o_1) \\ \\ z_0^0 & z_1^0 \end{bmatrix}$$

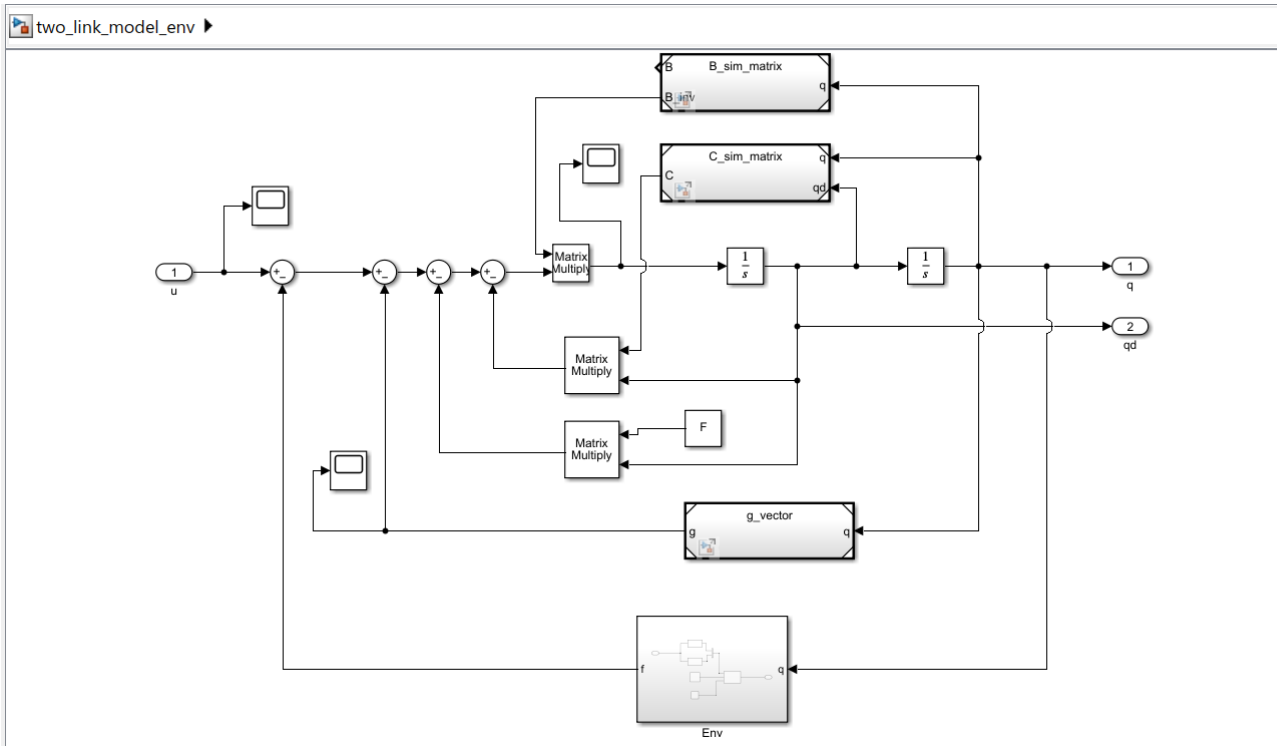
Where:

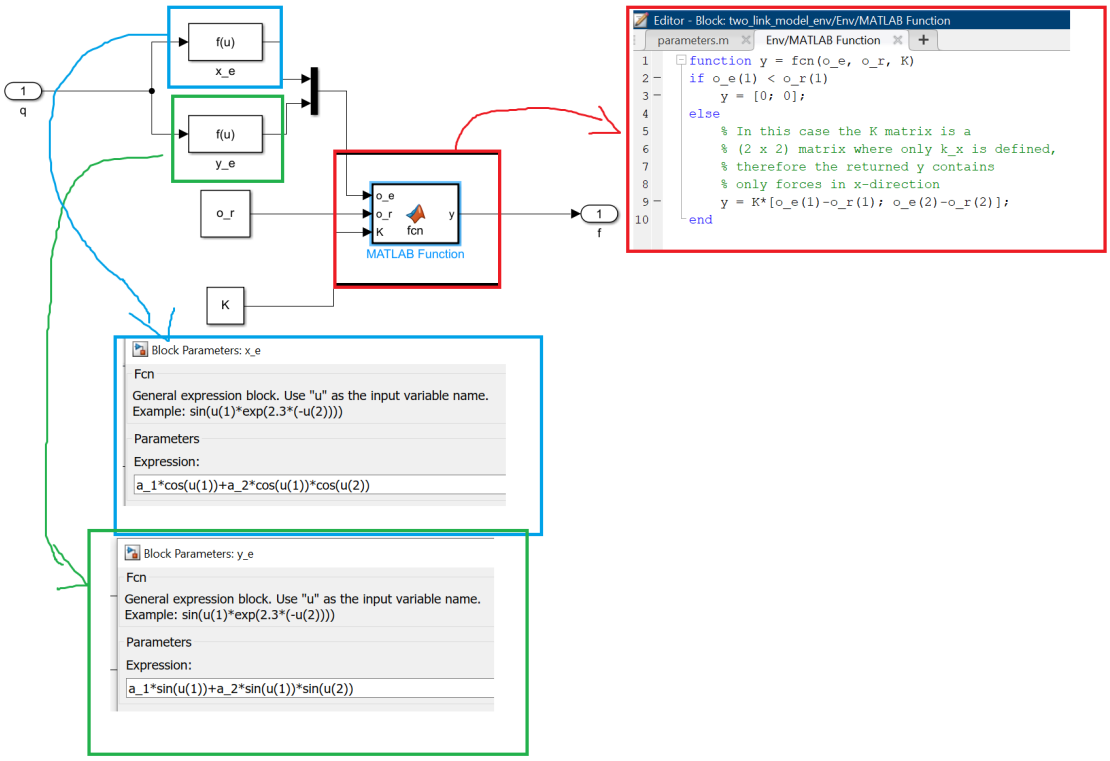
$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix}$$

$$z_0^0 = z_1^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J(q) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$


two_link_model_env
B_sim_matrix
B
B_inv
q
C_sim_matrix
C
qd
Matrix Multiply
Matrix Multiply
Matrix Multiply
F
g_vector
g
1
u
1
s
1
s
1
q
2
qd
f
q
Env



1.2 B

Given the operational state space $x_e = \begin{bmatrix} x_e \\ y_e \\ \phi \end{bmatrix}$ The geometric Jacobian can be reduced to $J(q) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 1 & 1 \end{bmatrix}$

By removing the elements describing velocity in z-direction, and rotational velocity in x- and y-direction

Given $v_e = J(q)\dot{q}$ and $\dot{x}_e = J_A \dot{q}$

that $v_e = \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$ and $\dot{x}_e = \begin{bmatrix} \dot{x}_x \\ \dot{x}_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$

$J_A = J$

1.3 C

Removing the rotational component from the Jacobian yields: $J = J_A = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$

In cases when

$$k_{Px}/k_x \gg 1, x_e \approx x_d$$

and opposite when

$$k_{Px}/k_x \ll 1, x_e \approx x_r$$

I.e. when the stiffness coefficient of the environment is much smaller than the stiffness coefficient of the manipulator, the manipulator end-effector reaches the equilibrium in the desired position and in opposite cases the end-effector reaches equilibrium at the rest position of the environment.

The parameters had the following values:

$$x_r = 0.5$$

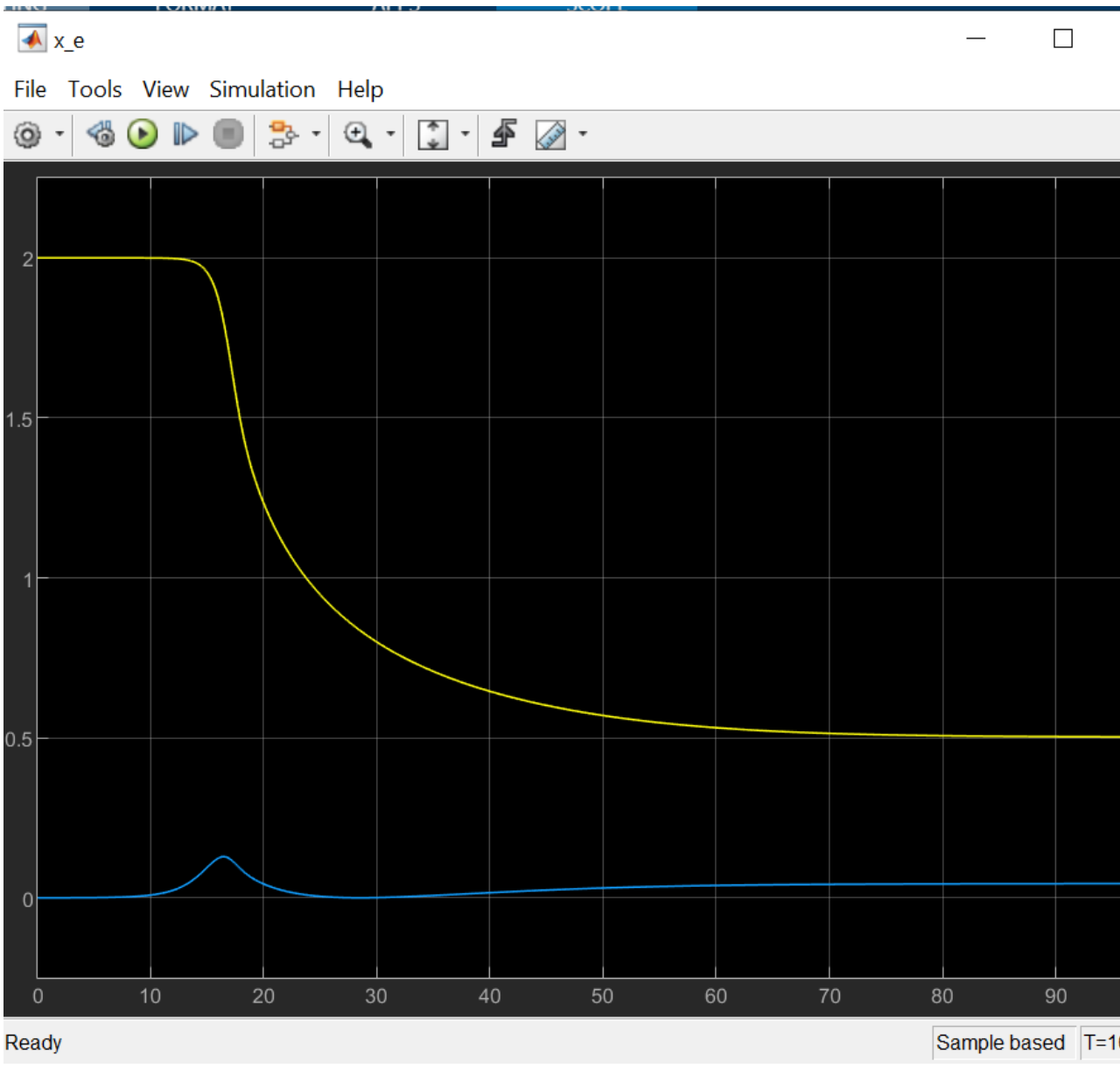
$$x_d = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$$

$$K_x = diag\{100, 0\}$$

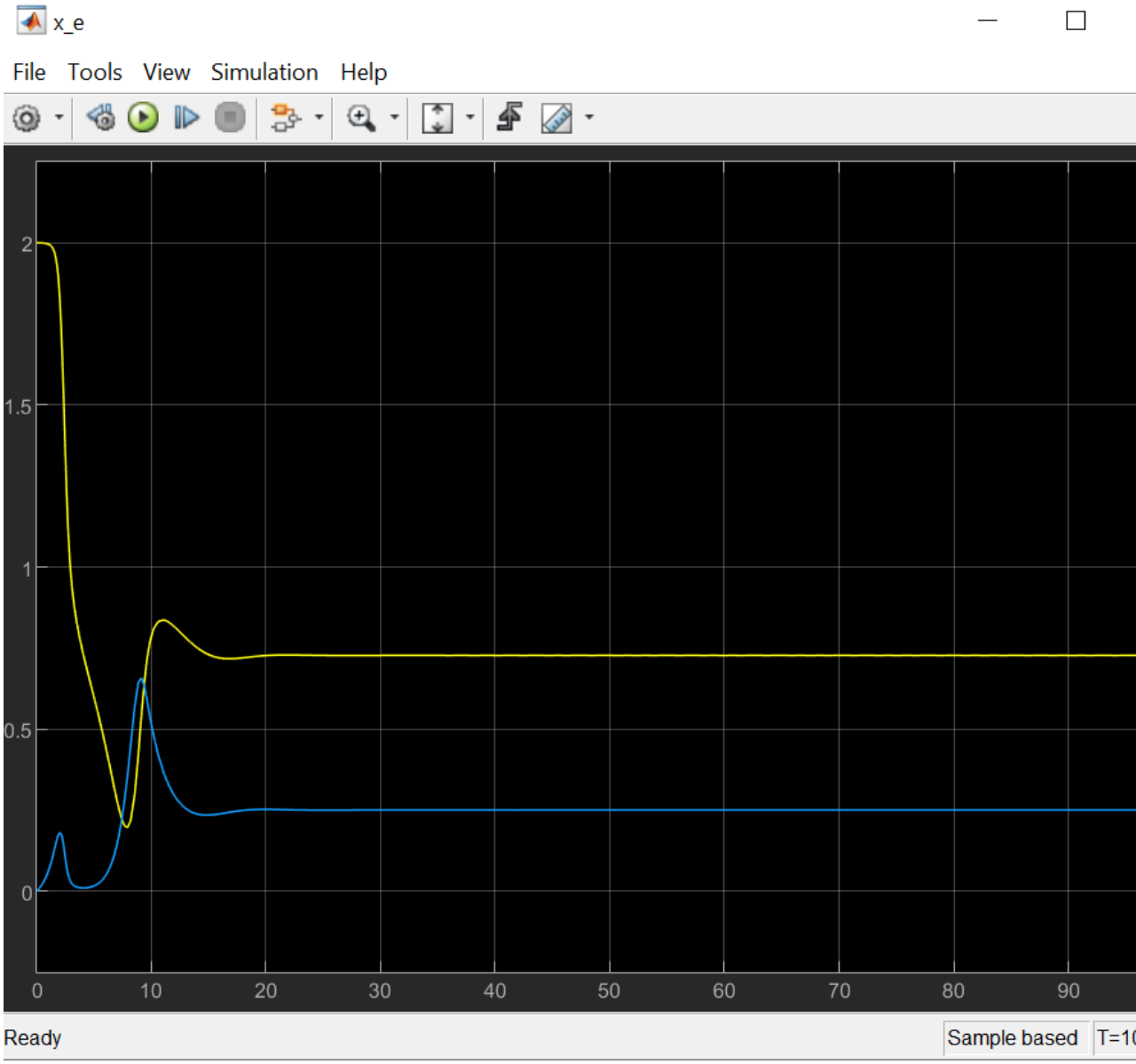
and the derivative term was set to 2000 as that gave readible resuslts:

$$K_D = diag\{2000, 0\}$$

Firstly, the K_P was set to 1, and the expected results were that the end-effector position settled at $x_r = 0.5$:


x_e
File Tools View Simulation Help
2
1.5
1
0.5
0
0 10 20 30 40 50 60 70 80 90
Ready
Sample based

Then the K_P was set to 1000, and the expected results were that the end-effector position settled at $x_d = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$:


x_e
File Tools View Simulation Help
2
1.5
1
0.5
0
0 10 20 30 40 50 60 70 80 90
Ready
Sample based
T=10

1.4 D

$$J_A = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$$

$$\dot{J}_A = \frac{d}{dt} J_A = \begin{bmatrix} -a_1 c_1 \dot{q}_1 - a_2 s_1 c_2 \dot{q}_2 - a_2 s_2 c_1 \dot{q}_1 & -a_2 s_1 c_2 \dot{q}_2 + s_2 c_1 \dot{q}_1 \\ \\ -a_1 s_1 \dot{q}_1 - a_2 s_1 c_2 \dot{q}_1 - a_2 s_2 c_1 \dot{q}_2 & -a_2 s_1 c_2 \dot{q}_1 + s_2 c_1 \dot{q}_2 \end{bmatrix}$$

Since the analytic and geometric Jacobian are equal, the transformation matrix between them is the identity matrix. Given the relationship

$$T_A^T h_e = h_A$$

and

$$T_A^T = I_3$$

then

$$h_e = h_A$$

h_A can also be expressed as $K_P \tilde{x}$

1.5 e) and g)

The simulation got errors, I think it might be due to the manipulator reaching a singularity

2 Task 3

2.1 a

Image-based visual servoing: Given a fixed object in the base frame, a number of characteristic features can be described. A desired pose of the camera is found by formulating a vector, s_d, of object features with certain values. The desired camera pose is described by

$$s_d = s(x_{d,o}$$

, where $x_{d,o}$ is the desired pose (meaning that when the camera is in the desired pose, the features appear in the image the desired manner). If the features are not positioned as desired in the image, the camera moves to reach the desired position. So if the object moves, the camera follows. The relationship between the pixel velocities (the velocities in which the pixels in the frame has for correcting a wrongly positioned object) and the camera velocities is the image Jacobian

Position-based visual servoing: Visual measurements are used to estimate the real-time homogeneous transformation matrix T_o^c. This matrix represent the relative pose of the object frame wrt. the camera frame. A desired pose

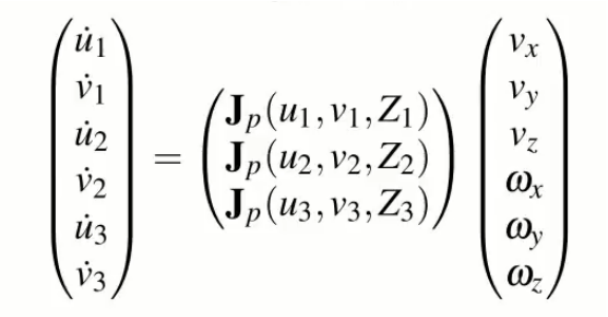


Figure 1: Image Jacobian for three points

of the object frame wrt the camera frame T_o^d, where d represent the desired frame of the camera, is specified. By using the matrix T_o^c (found by visual measurements) and the desired pose specified, T_o^d, the transformation matrix from the actual camera frame to this desired pose can then be specified,

$$T_c^d = T_o^d (T_o^c)^{-1} = \begin{bmatrix} R_c^d & o_{d,c}^d \\ 0^T & 1 \end{bmatrix}$$

The error between the desired pose and the camera pose is defined as

$$\tilde{x} = - \begin{bmatrix} o_{d,c}^d \\ \phi_{d,c} \end{bmatrix}$$

where the $\phi_{d,c}$ is extracted from the rotation matrix R_c^d. The controller should be designed so that the error tends to zero.
Position-based visual uses the geometric model of the object and camera specifications to estimate the transformation matrix of the object pose wrt the camera pose. The desired pose of the object wrt the camera is specified, and the error in pose is calculated. This error is then used to correct the pose of the camera to match the desired pose. Image-based visual servoing uses features in the object to position the camera so that the features are represented in a desired way in the camera frame. The position of the features depends on the relative pose between the camera and the object, but it directly computes the desired velocities of the camera that move the features to the desired place instead of using transformation matrices.

2.2 b

Visual measurements have a much lower update rate than what is desired in motion control of robot manipulators. The sampling rate will need to be adjusted to accommodate to this and preserve stability. A solution would be to have two levels of control: 1. High gain motion controller in joint or operational space

2. Visual servoing controller with lower sampling rate on top of the motion controller.

The motion controller is considered ideal (the output equals the reference) and the visual servoing control is achieved by computing the trajectory using the Jacobian.

This is called resolved velocity control because the control scheme is based on the computation of the joint velocities from the operational space error.

2.3 c

The image Jacobian relate the image feature velocities to the camera velocity (maps the velocities of the object wrt the camera, to the velocity of the image features). That we know how to move the image features in the camera frame can then be used to control the manipulator The relationship between the feature velocity and the velocity of the object wrt the camera is as follows:

$$\dot{s} = J_s(s, T_o^c) v_{c,o}^c$$

where $J_s(s, T_o^c)$ is the image Jacobian.
The interaction matrix describes the linear mapping between the part in the equation above related to the absolute velocity of the camera, v_c^c and the image plane velocity, $\dot{s}$. It is derived by expressing $v_{c,o}^c$ in terms of v_c^c and v_o^c and inserting the expression into $\dot{s} = J_s(s, T_o^c) v_{c,o}^c$.

$$v_{c,o}^c = \begin{bmatrix} \dot{o}_{c,o}^c \\ R_c^T(\omega_o - \omega_c) \end{bmatrix}$$

where

$$\dot{o}_{c,o}^c = R_c^T o_o - R_c^T o_c + S(o_{c,o}^c) R_c^T \omega_c$$

($S(o_{c,o}^c)$ is a skew symmetric matrix from the vector $o_{c,o}^c$, as per

$$v = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T \rightarrow S(v) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

and S(v)R is equivalent to taking the cross-product of v and R if R is a vector) This gives:

$$v_{c,o}^c = \begin{bmatrix} \dot{o}_{c,o}^c \\ R_c^T(\omega_o - \omega_c) \end{bmatrix} = \begin{bmatrix} \dot{o}_{c,o}^c \\ R_c^T \omega_o - R_c^T \omega_c \end{bmatrix} = \begin{bmatrix} R_c^T o_o - R_c^T o_c + S(o_{c,o}^c) R_c^T \omega_c \\ R_c^T \omega_o - R_c^T \omega_c \end{bmatrix}$$

Since

$$v_o^c = \begin{bmatrix} \dot{o}_o \\ R_c^T \omega_o \end{bmatrix}$$

and

$$v_c^c = \begin{bmatrix} \dot{o}_c^c \\ R_c^T \omega_c \end{bmatrix}$$

This gives

$$v^c_{c,o} = v^c_o + \begin{bmatrix} -R^T_c o_c + S(o^c_{c,o}) R^T_c \omega_c \\ R^T_c \omega_c \end{bmatrix} = v^c_o + \begin{bmatrix} -I & S(o^c_{c,o}) \\ 0 & I \end{bmatrix} \begin{bmatrix} R^T_c o_c \\ R^T_c \omega_c \end{bmatrix} = v^c_o + \begin{bmatrix} -I & S(o^c_{c,o}) \\ 0 & I \end{bmatrix} v^c_c$$

The matrix $\begin{bmatrix} -I & S(o^c_{c,o}) \\ 0 & I \end{bmatrix}$ is denoted $\Gamma(o^c_{c,o})$ and by inserting into the equation $\dot{s} = J_s(s, T^c_o) v^c_{c,o}$, we get

$$\dot{s} = J_s(s, T^c_o)(v^c_o + \Gamma(o^c_{c,o}) v^c_c) = J_s(s, T^c_o) v^c_o + J_s(s, T^c_o) \Gamma(o^c_{c,o}) v^c_c$$

where

$$J_s(s, T^c_o) \Gamma(o^c_{c,o})$$

is the interaction matrix mapping the camera velocity to the feature velocities, and in the case of a stationary object wrt the base frame, the interaction matrix is a linear mapping between the camera velocity and the feature velocities

3 4

3.1 a

Unilateral teleoperation means one robot is sending data to the other without receiving anything. Bilateral means the robots are sending data to each other. The data can for example be force feedback, where in the bilateral case the interaction forces between the manipulator follower and the environment is reflected back to the operator by acting on the master device. In a unilateral system, the operator would not get the force feedback from the follower.

3.2 b

The hybrid matrix represent the relationship between the forces and velocities in the master and follower robot, given as such:

$$\begin{bmatrix} f_m \\ -v_f \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} v_m \\ f_f \end{bmatrix} = H \begin{bmatrix} v_m \\ f_f \end{bmatrix}$$

where the H is the hybrid matrix, f_m is the master force, f_f is the follower force, v_m is the veloccity of the master and v_f is the velocity of the follower.
A transparent teleoperation is achieved when the forces and velocities in the follower and master robot are the same. The hybrid matrix is then skew-symmetric. This is ideal as it means no information is lost and the forces reflected on the operator corresponds exactly to the forces inflicted on the follower:

$$H = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The forward-flow teleoperation controller is a bilateral controller. Using force sensors isn't always a viable option as the sensors need to be sufficiently small

and yet have adequate measurement range when mounting them on the tip of the manipulator. The sensors also have noise and limited bandwidth. Therefore the forward-flow controller uses an estimation of the interaction force instead of actual measured force. The controller only communicate position-related measurements, as the interaction force is a function of position, velocity and acceleration of the robot. The acceleration and position are derived from the velocity.